Lab Week 6: Dynamic Programming

1. Design a dynamic programming algorithm for the “coin row problem” but change the constraint to be: You may not pick up a new coin until you have passed up at least two coins rather than one coin as in the problem discussed by the TA’s. Thus you goal is pick up the maximum amount of money picking up coins leaving at least two coins between each picked up coin.

Example 1 2 10 5 2 3 11 1 1 1 1 12 -- the answer would be to pick up coins 3(10), 7(11), 12(13) for a total of

a. Write the recurrence relation that will solve this version of the coin row problem.

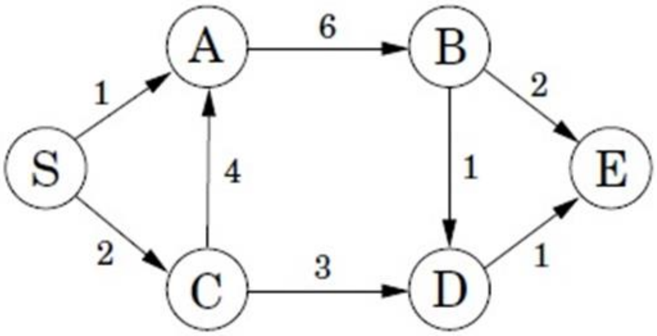
   Recall the recurrence relation for the original coin row problem was:

   $F(n) = \max\{c_n + F(n-2), F(n-1)\}$ for $n > 1$ $F(0)=0$ and $F(1)=c_1$

   Where c_n is the value of the coin in the n-th position

b. Fill in the table to solve the above example using your recurrence relation.

2. Find the **longest path from S to E** in the graph below using a dynamic programming similar to the one used to find the shortest path in a DAG



3.

4. Design a dynamic programming algorithm for the following problem. Find the maximum total sale price, MTSP, that can be obtained by cutting a rod of n units long into integer-length pieces if the sale price of a piece i units long is $\mathbf{p_i}$ for i = 1, 2, …, n .

Examples

- Rod Length 4, $p_1 = 2$, $p_2 = 4$, $p_3 = 7$, $p_4 = 8$ then MTSP = 9 sell lengths (1, 3)
- Rod Length 4, $p_1 = 3$, $p_2 = 7$, $p_3 = 9$, $p_4 = 12$ then MTSP = 14 sell lengths (2, 2)
- Rod Length 4, $p_1 = 2$, $p_2 = 4$, $p_3 = 7$, $p_4 = 11$ then MTSP = 11 sell lengths (4)
- Rod Length 4, $p_1 = 3$, $p_2 = 5$, $p_3 = 8$, $p_4 = 11$ then MTSP = 12 sell lengths (1, 1, 1, 1)

The following is a rough guide to the thought process (not to be submitted) you should follow:

1. Write an English description of what you are trying to optimize along with what the parameter represents: e.g. MTSP (k)
2. Write a recurrence relation for the objective function: MTSP (k) = some function involving smaller versions of the problem. Specify the base case. This specifies the optimal substructure of the problem.
3. Determine the table you will use to keep track of the MTSP(k)? The parameter(s) in the recurrence relation most likely will determine the dimensions.
4. Fill in the table and trace back for a simple version of the problem.

5. Write pseudo code snippets to fill in the table and then to trace back using the table to find the rod sizes that give you the optimal solution.
6. Implement your solution in java.

Submit your program to PolyLearn: A Java class *RodCutter.java*. Your program should read from a text file: “rodOptTest.txt”. The text file will contain a set of problems for your program to solve, see below. The output must be as shown!

Note: The output shows the contents of the table that is used in your dynamic programming solution to the given problem, then shows an optimal set of rod lengths.

As always, be sure to test you program before submission

INPUT FILE will not contain comments such as those shown below !!

```
2                                   // 2 test cases
10                                  // first case rod of length 10
2 4 4 5 12 13 14 15 40 41           // $p_1$ = 2, $p_2$=4 etc
16                                  // second case rod of length 16
1 4 6 25 28 31 80 81 82 83 84 85 86 88 90 92
```

```
Output:
Case 1:

Case 1:                                         Case2:
total for length 1        = 2                   total for length 1        = 1
total for length 2        = 4                   total for length 2        = 4
total for length 3        = 6                   total for length 3        = 6
total for length 4        = 8                   total for length 4        = 25
total for length 5        = 12                  total for length 5        = 28
total for length 6        = 14                  total for length 6        = 31
total for length 7        = 16                  total for length 7        = 80
total for length 8        = 18                  total for length 8        = 81
total for length 9        = 40                  total for length 9        = 84
total for length 10       = 42                  total for length 10       = 86
Optimal rod cutting                             total for length 11       = 105
Number of rods of length 1      = 1             total for length 12       = 108
Number of rods of length 9      = 1             total for length 13       = 111
                                                total for length 14       = 160
                                                total for length 15       = 161
                                                total for length 16       = 164

                                                Number of rods of length 2      = 1
                                                Number of rods of length 7      = 2
```